

United States Patent Application

for

METHOD AND SYSTEM FOR IMBEDDING XML FRAGMENTS IN  
XML DOCUMENTS DURING RUN-TIME

Inventor:

Andrew S. Nielsen

EXPRESS MAIL CERTIFICATE OF MAILING

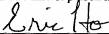
"Express Mail" mailing label number: **EV074663687US**

Date of Deposit: **January 22, 2002**

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

ERIC HO

(Typed or printed name of person mailing paper or fee)



(Signature of person mailing paper or fee)

METHOD AND SYSTEM FOR IMBEDDING XML FRAGMENTS IN XML  
DOCUMENTS DURING RUN-TIME

5 FIELD OF THE INVENTION

The present invention relates generally to the testing of Web services, and more particularly, to a method and system for imbedding XML fragments in XML documents during run-time.

10 BACKGROUND OF THE INVENTION

With the explosive growth of business to business (B2B) eCommerce, the Internet presents incredible opportunities for businesses of all sizes. For example, business to business (B2B) eCommerce provides opportunities to find new customers, to streamline supply chains, to provide new services, and to secure  
15 financial gain.

Organizations that have moved their business online are already realizing significant economic and competitive gains, such as increased revenue, lowered costs, new customer relationships, innovative branding opportunities, and the creation of new lines of customer service.

20 Despite the outstanding growth of B2B eCommerce in the last few years, there exists a major impediment to opening up worldwide trade to those already conducting B2B eCommerce and to the businesses that are not yet players in the digital economy.

This impediment can be described as follows. Most eCommerce-enabling  
25 applications and Web services currently in place employ divergent paths to connect buyers, suppliers, marketplaces, and service providers. Without large investments in technology infrastructure, a semiconductor manufacturer in Taiwan, a furniture manufacturer in Pennsylvania, and a specialized industrial engineering firm in New

Delhi can transact Internet-based business only with the global trading partners they have discovered and, of those, only the ones using the same applications and Web services. As can be appreciated, this current model is restrictive and limiting.

In order to fully open the doors to these existing and potential B2B players, successful eCommerce requires that businesses be able to 1) discover each other, 2) make their needs and capabilities known, and 3) integrate services using each businesses' preferred technology, Web services, and commerce processes.

To address this challenge, a group of technology and business leaders have come together to develop the Universal Description, Discovery and Integration [UDDI] specification. The UDDI specification sets forth a global, platform-independent, open framework to enable businesses to (1) discover each other, (2) define how they interact over the Internet, and (3) share information in a global registry, thereby accelerating the global adoption of B2B eCommerce. UDDI is also a building block to enable businesses to quickly, easily and dynamically find and transact with one another via their preferred applications. Participation in UDDI can help an established B2B eCommerce player expand into new markets and services or allow a company, who is new to the online space, to accelerate toward a world-class business presence.

The UDDI specifications take advantage of World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) standards such as Extensible Markup Language (XML), HTTP, and Domain Name System (DNS) protocols. Additionally, cross platform programming features are addressed by adopting early versions of the proposed Simple Object Access Protocol (SOAP) messaging specifications found at the W3C Web site.

The UDDI specification envisions distributed Web-based information registries of Web services. These UDDI registries are used to promote and discover distributed Web services. UDDI is also a publicly accessible set of implementations

of the specification that allow businesses to register information regarding Web services they offer so that other businesses can find them.

In the development of web servers that implement UDDI services, there is a need for mechanisms to test these services in order to ensure the proper operation thereof. Typically, a test suite file is written that tests the various operations supported by the Web server.

Unfortunately, the testing of web services poses several significant problems. One of these problems that are faced by testers of XML documents based web services is that often the information returned from a request cannot be determined at the time the tests are created. For example, the information may only be known at run-time after the server to be tested returns the information.

For example, a web service may support the saving of some object. The service often assigns the object a key, a tracking number, or other run-time values. The service also provides a way to look up the object (e.g., retrieve the information in the future). The testing of this service can employ a test suite file that saves the item in the first step, and when successful, looks up the previously saved item in the second step. This second step ensures that the save step executed properly.

One challenge to those devising appropriate test suite file is how to write a static test case that captures information that is not known at the time the test suite is created. This challenge is essentially a problem of timing. At the time one writes the test suite file, the run-time values of the object do not exist. The run-time values are assigned when the test is run (i.e., at run-time).

Accordingly, it would be desirable for there to be a mechanism for dynamically modifying the test suite file during runtime with data unavailable when the test suite file is created. It is also desirable for there to be a mechanism that allows the test creator to specify a location in the test suite file for inserting the data.

Based on the foregoing, there remains a need for a method that dynamically modifies a test suite file during runtime with data that is unavailable when the test

suite file is created, that allows a test creator to specify where in the test suite file to insert the data, and that overcomes the disadvantages set forth previously.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219

SUMMARY OF THE INVENTION

According to one embodiment of the present invention, a method for extracting data from runtime test results and for injecting the extracted data into subsequent verification calls is described.

5        According to one embodiment, a mechanism for modifying a mark-up language document (e.g., an XML test suite file) at run-time with data unavailable when the mark-up language document is created. The mechanism of the present invention, for example, extracts data from runtime test results (e.g., response from a server under test) and injects the extracted data into a portion of the test suite file that  
10        may be specified by the user. For example, the extracted data may be utilized in subsequent verification calls.

Other features and advantages of the present invention will be apparent from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements.

5        FIG. 1 illustrates a test infrastructure in which the mechanism of the present invention may be implemented.

FIG. 2 is block diagram illustrating in greater detail the dynamic markup language document modification module of FIG. 1 in accordance with one embodiment of the present invention.

10        FIG. 3 is a flow chart illustrating the steps performed by the dynamic markup language document modification mechanism of FIG. 1 for processing interactions in accordance with one embodiment of the present invention.

FIG. 4 is a flow chart illustrating the steps for replacing references in accordance with one embodiment of the present invention.

15        FIG. 5 illustrates an exemplary portion of a test suite file that includes interaction elements according to one embodiment of the present invention.

FIG. 6 illustrates the interaction element of FIG. 5 after an actual response has been added thereto according to one embodiment of the present invention.

20        FIG. 7 illustrates the test suite file of FIG. 6 after an expected response element has had its references replaced with a key value obtained at run-time according to one embodiment of the present invention.

FIG. 8 illustrates the test suite file of FIG. 7 with the actualResponse element added to the second interaction element.

25        FIG. 9 illustrates the test suite file of FIG. 8 with its internal representation changed to reflect the key attribute value obtained at run-time.

DETAILED DESCRIPTION

A method for modifying a mark-up language document (e.g., an XML test suite file) at run-time with data unavailable when the mark-up language document is created is described. For example, the method of the present invention can be employed to extract data from a run-time environment and to inject the extracted data into another portion of software code (e.g., a subsequent verification call). In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

It is noted that aspects of the present invention are describe in connection with a document (e.g., a test suite file) in the Extensible Markup Language (XML). However, it is to be appreciated that the teachings of the present invention extend to other documents that are described with other mark-up languages.

Test Infrastructure 100

FIG. 1 illustrates a test infrastructure 100 in which the mechanism of the present invention may be implemented. The test infrastructure 100 includes a test suite file 110 that is written in a markup language (e.g., the Extensible Markup Language (XML)). The test suite file 110 includes software code for testing one or more functions of a server 160 (e.g., a UDDI server) under test. The test suite file 110 includes at least one set of insertion tags 118, which are described in greater detail hereinafter.

The test infrastructure 100 also includes a dynamic markup language document modification module (which is referred to herein as dynamic document modification module or mechanism (DDMM)) 120. The dynamic markup language



document modification module 120 modifies the mark-up language document 110 (e.g., an XML test suite file) at run-time with data 144 that is unavailable when the mark-up language document 110 is created. The insertion tags 118 may be used by a creator of the test suit file 110 to specify a location in the document for inserting the data 144.

It is noted that these changes or modification are made to the representation 134 (i.e., the original XML test suite file 110 is not modified by the present invention). Furthermore, after execution, the additions or changes to the internal representation 134 of the markup language need not be saved, but may be saved in the case of UDDI. The dynamic markup language document modification module 120 is described in greater detail hereinafter with reference to FIG. 2.

The test infrastructure 100 also includes a run-time environment 140 that generates the data 144. For example, the run-time environment 140 can generate a request 164 that is directed to the server under test 160. In response, the server under test 160 generates a response 168 that may include data 144.

One example of the data 144 is a key. A key may be any unique identifier for identifying a particular object in the server (e.g., an identifier for a newly save business object or entity). It is noted that other attributes or other data may be extracted from the actual response element and injected to other portions of the test suite. These other data can include, but is not limited to, an operator name that specifies the name of the operator of the UDDI server, an authorized name that specifies the name of a publisher (e.g., a publisher that registers with a UDDI server). Examples of the data 144, which may be extracted from the response 168, are described in greater detail hereinafter.

The dynamic markup language document modification module 120 includes a parser 130 for receiving a markup language document (e.g., an XML test suite file 110) and based thereon for generating a representation 134 (e.g., an internal representation) of the document 110. For example, this internal representation 134

may be a document object model (DOM), which is a tree like data structure representation of an XML document.

The DDMM 120 also includes an injection mechanism 140 for receiving the data 144 from the run-time environment 140 and injecting the data 144 into the representation 134. The injection mechanism 140 is described in greater detail hereinafter with reference to FIG. 2.

In one embodiment, the mechanism of the present invention is implemented within an XML test infrastructure. For example, in testing UDDI servers, "save" calls return the same form of information that is returned by the "get" calls. To test whether a "save" request is successful, one first performs a "save" request followed by a "get" request. In this manner, the information that is saved by UDDI server in response to the "save" request may be compared to the information provided by the server in response to a "get" request. In this manner, the proper operation of the "save" request may be verified.

#### DDMM 120

FIG. 2 is block diagram illustrating in greater detail the DDMM 120 of FIG. 1 in accordance with one embodiment of the present invention. Specifically, FIG. 2 illustrates in greater detail the injection mechanism 140 of the DDMM 120. The injection mechanism 140 receives the representation 134 of the document and the data 144 from the run-time environment 140. Based on these inputs, the injection mechanism 140 generates a modified representation 138 of the document that is sent to the run-time environment for further execution.

The injection mechanism 140 includes a node creator 210 for creating a new node with the data 144. The injection mechanism 140 also includes a node adder 220 that adds the newly created node to the location in the tree (e.g., internal DOM representation) specified by the location of the insertion tags 118. The injection

mechanism 140 also includes a node replacer 230 for removing the node that existed prior to the insertion of the new node.

#### Operation of a UDDI Server

5           An example of how web services are offered and found is now described. First, a provider of goods or services first registers with a server, such as a UDDI server. The provider, for example, registers with a UDDI server and uses a save request to save information regarding the business entity to the server. This information can include, the business entity, name of the business, contact person,  
10       contact information, a list of goods or services provided, a description of each of the goods or services, etc.

              Second, a consumer searches a UDDI server for business that may be able to provide a particular good or services. For example, a consumer can use a “find” request to receive a list of keys (e.g. globally unique identifier; GUID) of business  
15       that may be able to supply the good or service.

              Next, the consumer can then use a “get” request with a particular key to obtain further information about a particular business (e.g., a supplier).

              The consumer can then make contact with the supplier and utilize other E-commerce services, such as request for quote and purchase order services, to proceed  
20       with a business transaction.

#### Requests for UDDI Server

              Universal Description and Discovery Interface (UDDI) specifies an XML/SOAP based web service. One important aspect to test is to determine whether  
25       parties can successfully register with a UDDI server. In UDDI, there are four types of requests: 1) a find request, 2) a get request, 3) a save request, and a delete request. A find request obtains an abbreviated version of an object. A get request obtains a non-

abbreviated version of an object or entity. A save request saves an object to a server. A delete request deletes a specified object from the server.

During the execution of a save request, the server assigns the information being save a key. The key can be, for example, a globally unique identifier (GUID) that a server assigns to a saved object. When a save request leaves a GUID field blank, the server assigns a GUID to the object being save and returns the GUID in a response. The GUID is an example of data 144 that is not known at the time that the test file is written and that may be added or inserted into the internal representation 134 of the test file at run-time

#### Test Structure

Test suite files 110 are typically structured in the following way. A test file 110 includes three items: 1) a setup element, 2) an interactions element, and 3) a tear down element. The setup element and tear down element are well-known to those of ordinary skill in the art and are not described in greater detail hereinafter. The interaction section includes a plurality of interaction elements. In this example, the interaction includes a request, which is sent to the service, and an expectedResponse element, which is used to determine the success of the request.

The processing performed by the DDMM 120 according to one embodiment of the present invention is now described. First, a test request 164 is sent to a server under test 160. Second, a response 168 to test request is received from the server 160. Third, the representation 134 is modified by the response168. Preferably, the entire response is added to the representation 134.

As described previously, the DDMM 120 of the present invention modifies an internal representation 134 of the test file or suite. For example, the internal representation 134 may be a data structure (e.g., a XML tree) that represents the test suite.

Modifying the representation 134 with the response 168 can involve the sub-steps of 1) creating an actual response node, 2) filling the actual response node with data from the response 168, and 3) adding the actual response node to the representation (e.g., to a DOM XML tree). Fourth, a portion of the response 168 is  
 5 injected as specified by the reference tags 118 (e.g., insertion tags) by employ a reference technology. For example, this step can involve replacing the reference element with at least a portion of the actual response node.

Specifically, FIG. 3 is a flow chart illustrating the steps performed by the DDMM 120 of FIG. 1 for processing interactions in accordance with one  
 10 embodiment of the present invention. Prior to the steps illustrated in FIG. 3, the following steps may be performed. First, a markup document (e.g., an XML test suite file) is received and read by the parser 130. Second, the parser 130 generates an internal representation (e.g., a DOM representation) of the test suite file. The interaction processing, which is described in FIG. 3, and the replacement processing,  
 15 which is described in FIG. 4, are then performed on the internal representation 134.

#### Interaction Processing

For each interaction *x* in interactions, the following steps are performed. In step 310, the references in the current interaction's request are replaced with the thing  
 20 the references point to. This step is described in greater detail hereinafter with reference to FIG. 4. A technology, such as Xpath, may be utilized for this purpose. One aspect of the present invention is the use of a referencing technology, such as Xpath for referencing a portion of the same document (i.e., a portion internal to the document). In the prior art, the referencing technology has been generally limited to  
 25 referencing a portion of a document external to the document where the reference exists. For example, an Xpath reference in a first document is typically utilized to print a portion of a second document.

In step 320, the request of the current interaction is then sent to the service (e.g., UDDI service). In step 330, the response is received from the service. In step 340, the actualResponse element is created based on the response.

In step 350, the actualResponse element is added to the current interaction  
5 after the expectedResponse.

In step 360, the references in the expectedResponse of the current interaction are then replaced with the things to which the references point. This step is described in greater detail hereinafter with reference to FIG. 4.

In step 370, the expectedResponse is compared with the actualResponse.

When the compare fails, in step 380 the test fails, and the test is stopped. The  
10 comparison step may be performed by employing a variety of different techniques. Further details concerning particular embodiments of the present invention may also be found in the following copending patent application which was filed on the same date as this application and which is hereby incorporated herein by reference. The  
15 copending application is as follows: " METHOD AND SYSTEM FOR COMPARING STRUCTURED DOCUMENTS " by inventor Andrew Nielsen.

#### Replacement Processing

FIG. 4 is a flow chart illustrating the steps for replacing references in  
20 accordance with one embodiment of the present invention. For each attribute or element node in the XML tree, the following steps are performed. In step 410, a determination is made whether the current node or attribute includes a reference flag. For example, a determination may be made whether a node's name begins with a "ref:". When the current node or attribute includes a reference flag, processing  
25 proceeds to step 420. Otherwise, when the current node or attribute does not include a reference flag, processing proceeds to step 424 where the next node or attribute is processed.

In step 420, determine or find a target specified by the reference node.

In step 430, the reference node is removed from the XML tree. In step 440, the target is copied to the same location where the reference node previously existed.

For each attribute or element node in the XML tree, the following steps are performed according to one embodiment of the present invention.

- 5 When the node's name begins with "ref:" (this node is hereinafter referred to as the source), the following steps are performed. First, the replacement module finds what the attribute or element's XPath points to (which is referred to herein as a target). Second, the replacement module removes the source node and replaces the source node with a new node of the same type (e.g., attribute or element). The name  
10 for the node is the name of the source node with the "ref:" portion removed.

When both the source and target nodes are of the same type (i.e., when both the source node and target node are attributes or when both the source nodes and target node are both elements), the value of the new node is a copy of the value in the destination.

- 15 When the source is an attribute, and the target is an element; the source attribute takes on the value of the concatenation of all text nodes in the target element. When the source is an element, and the target is an attribute; the source element contains one text node that includes the same text value that is in the target attribute.

20

#### Exemplary Test Suite File

- FIG. 5 illustrates an exemplary test suite file that includes interaction elements. The test suite file is written in XML. XML consists of elements, attributes, and text. Examples of these are provided as follows: An empty element:  
25 "<TagName/>" or "<TagName></TagName>", an attribute in an empty: "<TagName AttrName='attr value'>", and an element containing text "<TagName>The text</TagName>".

An integral part of XML is its containment relationship. Elements contain attributes and other elements. In this example: "<Tag1 Attr1='value1'><Tag2/></Tag1>", the element "Tag1" contains an attribute "attr1" and an element "Tag2". Attributes contain only text values. There is no limit on the number of contained elements or the depth of containment. Attributes contained in an element are required to have unique names, but elements do not share this restriction.

An XML document has only one root element. There are also certain rules about how and where to use "<", ">", and "/" characters. Elements if they contain no text or other elements can have the form "<TagName/>" or "<TagName></TagName>". Elements that have contents must be of the form "<TagName>contents</TagName>". The first tag is called the beginning tag and the other tag is called the ending tag.

Tag names must match exactly according to character and case. Text may not contain "<" or "&" characters; if such characters are desired they need to be replaced with "&gt;" and "&amp;" respectively.

A walk-through of the processing performed by the present invention is now described. First, the test suite document is parsed into a DOM, which is a standard way to programmatically represent XML documents internal to a program. As described previously, the mechanisms of the present invention modify the internal representation of the document. Similarly, the modifications, replacements, and insertions in the exemplary test document illustrated in FIGS. 5-9 represent changes and modifications to the internal representation of the document and not to the document itself. It is noted that the test document is not modified by the mechanisms of the present invention.

In this example, there are two iterations since there are two interaction elements in the interaction section. The first interaction generates a save request, and the second interaction generates a get request.



Iteration 1 - Step 1

Since the save request has no "ref:" elements or attributes, no processing is performed in step 1.

5

Iteration 1 - Step 2

The request is sent to the server. The mechanism for sending requests to servers is well-known to those of ordinary skill in the art. In the case of UDDI, requests are sent using HTTP or HTTPS POST. The body of the POST contains a SOAP envelope and body that contains the contents of the request element. In this case, an exemplary request can be in the following form:

10

```
<save_object>
  <theObject>...</theObject>
</save_object>
```

15

20

Iteration 1 - Step 3

HTTP POST returns a response that includes data. This data can be in the form of a SOAP XML document containing a SOAP envelope and body. The body includes the actual response data, which is of interest.

25

Iteration 1 - Step 4

A new element is created and given the tag name "actualResponse". The "data of interest" from step 3 is added to this new element.

30

Iteration 1 Step 5

This new actualResponse element is then inserted into the document just below the expected response. FIG. 6 illustrates the test suite file of FIG. 5 with the actualResponse element added thereto.

5

Iteration 1 Step 6

In this iteration, this step is uninteresting since there are no ref: attributes or elements to replace.

10

Iteration 1 Step 7

Comparison of the two documents is now performed. A comparison algorithm may be applied to both of the documents. The comparison algorithm tests the two documents for equality and returns a failure if they are not equal.

15

Iteration 1 Step 8

If for some reason the comparison determines that the two documents are not equal, the test is stopped.

Iteration 2 Step 1

20

The algorithm for this step (as well as step 6) is defined in the algorithm called "replace references" described previously. The algorithm involves traversing the request element (e.g., the DOM tree) in order to search for nodes or attributes with names beginning with an insertion tag (e.g., "ref:"). When nodes or attributes with the "ref:" tag are found, the contents of such nodes or attributes are treated as an XPath relative to the current node. This XPath leads us to another node.

25

These nodes, as described above, are utilized to replace the "ref:" node with a new node based on the contents of the referenced node. FIG. 7 illustrates the test suite file of FIG. 6 after an expected response element has had its references replaced

with a key value obtained at run-time according to one embodiment of the present invention.

Iteration 2 Step 2

5 A request is sent over HTTP or HTTPS.

Iteration 2 Step 3

A response is received.

Iteration 2 Step 4

10 An actualResponse element is created.

Iteration 2 Step 5

15 The actualResponse element is added to the document. FIG. 8 illustrates the test suite file of FIG. 7 with the actualResponse element added to the second interaction element.

Iteration 2 Step 6

20 In this step, the algorithm of step 1 is performed except in this case we traverse the expectedResponse (e.g., DOM tree). FIG. 9 illustrates the test suite file of FIG. 8 with its internal representation changed to reflect the key attribute value obtained at run-time.

Iteration 2 Step 7

25 The actual and expected responses are compared.

Iteration 2 Step 8

If for some reason the comparison determines that the two documents are not equal, the test is stopped.

5           The principles of the present invention are described in the context of a real-time method for embedding XML fragments into XML documents. However, it is noted that the teaching of the present invention can be applied to any structured document and other applications. It is noted that other technologies may be utilized to reference another portion of the same structured document.

10           One advantage of the present invention is that the mechanism of the present invention allows a test file or suite to be written that refers to or uses information that is unknown until run-time.

Another advantage of the present invention is that test files or suites may be shortened by using a reference to another portion of the same test file instead of  
15   having to copy or duplicate the portion of same test code in multiple places in the same document. For example, since the results of both the "save" and "get" calls, in UDDI, are essentially the same, and the expected result is fully specified in the "save" call, the expected response in the "get" call can simply make a reference to the save's expected response by utilizing the mechanisms of the present invention. One  
20   advantage of this approach is that the test files may be simplified and shortened by using references to another portion of the test file in lieu of repeating or duplicating that portion of the test file (e.g., expected responses).

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various  
25   modifications and changes may be made thereto without departing from the broader scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

---